Bolt Beranek and Newman Inc.

12

Report No. 3736

Distributed Computation and TENEX-Related Activities:

Quarterly Progress Report No. 9, 1 November 1976 to 31 January 1977

December 1977

Prepared for:
Defense Advanced Research Projects Agency

BBN Report No. 3736

DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES

Quarterly Progress Report No. 9
1 November 1976 to 31 January 1977

December 1976

Sponsored by:

Defense Advanced Research Projects Agency
ARPA Order No. 2935

Monitored by:

Office of Naval Research
Under Contract No. N00014-75-C-0773
Contract Period 1 November 1974 to 1 May 1978

9 Quarterly prog[illegible] rept. no. 9,
1 Nov 76-31 Jan 77,

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) BBN [redacted]-3736	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) DISTRIBUTED COMPUTATION AND TENEX-RELATED ACTIVITIES.		5. TYPE OF REPORT & PERIOD COVERED 11/1/76 - 1/31/77
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) R. Schantz, R. Thomas		8. CONTRACT OR GRANT NUMBER(s) (15) NØØØ14-75-C-Ø773, ARPA Order-2935
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 5Ø Moulton Street Cambridge, Massachusetts Ø2138		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE (11) Dec[redacted]76
		13. NUMBER OF PAGES ~~29~~ (12) 33p.
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES
This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2935.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
distributed computation
National Software Works
distributed operating system
TENEX operating system

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
This report describes ~~BBN~~ efforts in the design of the National Software Works system and ~~BBN~~ efforts to integrate TENEX into the National Software Works system.

Ø6Ø1ØØ

[illegible]

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

1. Introduction

During this quarter, the National Software Works (NSW) development effort progressed in a variety of areas. An important milestone was the project review and system demonstration held in December, 1976 at the Information Sciences Institute of the University of Southern California. The review and demonstration, including the first demonstration of the tools for the UYK-20 programming environment, served to emphasize the significant progress made in the project over the last half year. However, the system remains as just a shell for the prototype implementation, with much to be done in terms of achieving operational status and a significant functionality.

In the following sections we discuss this quarter's major design and implementation efforts by BBN toward achieving an operational NSW. The design and implementation aspects of our work focus primarily on the MSG (BBN Report 3237) and Foreman (BBN Report 3266) NSW components and their realization for the TENEX family of computer systems. Additionally, we discuss the installation of new TENEX based NSW tools, and negotiations which are currently underway with DEC for support of the pending development of TOPS-20 as a Tool Bearing Host. Finally, we report on the various NSW meetings and seminars held during the three month period ending in January 1977.

2. MSG: The NSW Interprocess Communication Facility

This quarter we have prepared and distributed an updated version of the MSG design specification document. The updated report (BBN Report 3483) incorporates all of the design changes and clarifications resulting from the collective experience of building the initial versions of NSW, including three different host MSG implementations. Newly documented base MSG features include an MSG primitive which provides a process with its own complete MSG name, and a facility for alerting a process to a spontaneous break in one of its MSG direct connections. The revised document also now contains appendices outlining a scenario for NSW use of MSG and a series of detailed implementation notes and suggestions, including a standardized set of MSG error codes. Also appearing for the first time is a state machine description for handling MSG messages. The finite state machine concept is used as a concise model for providing greater descriptive detail on using the MSG-to-MSG protocol to support interprocess messages. This form of specification has already proven extremely useful and convenient in guiding new MSG implementations.

In addition to our MSG design and documentation efforts, we have also progressed in the TENEX MSG implementation area. The major TENEX MSG implementation efforts for this quarter were in the three areas of functional improvements and modifications, additional operational aids, and performance measurement. These are discussed individually in the following sections.

2.1 Functional Improvements

Since the initial NSW was almost exclusively TENEX based, not all parts of the MSG protocol were required to be implemented in the early released versions. In particular, since TENEX MSG does not rely on MSG based inter-computer message flow control, it had no need to immediately be responsive to these protocol commands as an operational prerequisite. However, as other MSG implementations were completed, this assumption was no longer valid. Hence, MSG flow control was added to the TENEX MSG implementation during the last quarter.

MSG flow control permits a receiving MSG host to request that the sending MSG host buffer the message data until it is called for by the receiving host. It's major emphasis is in trying to alleviate some of the problems resulting from limited storage facilities typically found on small host computers and non-virtual memory machines. In effect, a small host can use the storage area of the larger hosts to buffer messages originating in these larger hosts. Then, when the message data is actually needed, it can be retrieved from the buffering host. Since the TENEX operating system provides ample secondary storage as well as a large virtual address space in which to manage messages, the

TENEX MSG does not request that other hosts buffer its messages. However, as a result of our flow control implementation, TENEX MSG can now provide this buffering capability to other more limited NSW hosts.

MSG has undergone an internal structural change which should be transparent to its users but is aimed at increasing our ability to reconstruct the events leading to certain failures. To accomplish this, we have changed the internal buffer allocation strategy to use circular buffers for process control storage blocks, job control blocks, pending event blocks, etc. As a result, the buffer most recently deallocated will be the last to be reallocated. The importance of this change is that it preserves an audit trail through many of the previous MSG internal state configurations, even after they become obsolete. Thus, a partial state transition backward through time is maintained for the longest possible interval without requiring extra resources to be dedicated to this function. The retained data proves invaluable in postmortem investigations of internally and externally detected failures, and has helped uncover a number of system bugs.

In an implementation effort that has been carried over from the last reporting period, we have successfully integrated the MSG user Telnet code into the Compass NSW front end component. This code, developed last quarter, provides a program interface to the user Telnet protocol function and can be used in conjunction with MSG primitive operations for obtaining direct connections of type "user Telnet". After an initial integration effort, the Telnet code became operational within the FE and is now used as the regular communication support for user tool sessions (See also Section 3.3).

2.2 Operational Aids

We have previously reported on the development of an MSG process monitoring and debugging capability supporting a variety of commands. This quarter we have added to the list of available commands in providing an augmented debugging and event monitoring environment. There are new commands for setting minimum timeout intervals, for forcing the timeout of a pending event, and for controlling the logging of MSG process events.

An important concept in MSG is that of a pending event. When a primitive operation is issued which cannot be immediately completed, a pending event is created and control can be returned to the issuing process. When the operation ultimately completes, MSG "notifies" the process of the "completion of the pending event." Completion of an operation often will require the cooperation of some other process. For example, in order to receive a message, some other process must send one. To avoid the possibility of waiting indefinitely for a pending event, each has a timeout interval which when elapsed will automatically cause the event to be immediately signalled with an appropriate error indicator. Each component declares the appropriate timeout interval along with each of its MSG operations. The timeout interval is generally set to approximate the expected delay to complete the operation, taking into account network delays and system scheduling as well as function execution times. When a pending event is timed out, the signalled process generally assumes an error has occurred and either takes corrective action or terminates. During a system testing and debugging session, one often needs finer control over the timeouts without necessitating the actual modication of the programs that utilize timeout intervals. For example, when debugging, long delays are usually associated with breakpointing a running program. Thus it is important to be able to set extended timeout intervals in some components when debugging another component. The MSG command MINIMUM allows the configuration manager to set appropriately large values as the shortest interval which triggers a timeout. Additionally, the MINIMUM command can be a tool with which a configuration operator can adaptively set the timeout values to more accurately reflect the current system and network delays in handling NSW operations. By judiciously selecting the timeout interval to match the current system configuration state, the unproductive work associated with a premature timeout can oftentimes be avoided.

Another new command which is extremely useful in testing as well as in operational situations is the FORCE command. Using this facility, an NSW operator can force a pending event to be immediately removed and reported as timed out. In addition to the obvious utility in testing a program's timeout log ', the FORCE command can be operationally used to hasten the cleanup and recovery operations associated with erroneous or unforeseen component behavior. Because of the rather complicated nature of the NSW component interactions, the procedures for reobtaining a consistent system state after a malfunction will frequently tie up resources for a substantial interval. When such a malfunction is recognized by an operator, the FORCE command can be used to immediately begin the process of stabilizing the state of the effected components.

Commands have also been added for event logging (LOGGING) and MSG performance measurement (STATISTICS). These are part of the larger NSW wide system performance evaluation and improvement efforts reported in the next section.

2.3 Performance Measurement

The project review meeting held this quarter included a working demonstration of an anticipated typical NSW usage pattern. The demonstration clearly showed the significant progress of the system development effort and reinforced the validity of the NSW concept. However, it also clearly showed that the current system organization and components do not provide a level of performance which would be acceptable in an operational system. Since an operational NSW is a high priority item over the next year, it is apparent that performance measurement and evaluation, along with any appropriate system modifications, will also be high priority tasks.

This quarter, we have begun our performance evaluation effort. There will be two major aspects to the performance measurement task. One aspect is to completely instrument and subsequently evaluate (and modify) those TENEX NSW components which we have implemented. This effort is intended to measure the behavior of the algorithms and strategies used internally by the components, and to identify any structures significantly contributing to the poor NSW response characteristics. The other aspect of the performance effort is to try to capture data which provides an accurate picture of NSW behavior as an integrated system. From our vantage point within the operating system and within our NSW components, we plan to provide adequate instrumentation for an NSW running given scripts. From this we hope to be able to account for the lengthy elapsed time in handling typical NSW operations and to identify serious system bottlenecks. We believe these measurement and evaluation efforts to be a necessary first step in proposing and implementing performance improvement measures.

As previously mentioned, we have started our measurement effort by introducing some preliminary instrumentation facilities into TENEX MSG. MSG records the events associated with handling NSW processes and their MSG primitive operations. This facility is controlled by the LOGGING command, and produces a history file with timestamped entries denoting the MSG observable activities. The history file not only provides a recording of the component operations employed in implementing a particular function, but also is viewed as a means of approximating the service time for these functions. During the next quarter, we hope to be able to develop software to process these log files, and to report on our observations.

We have also introduced some amount of internal MSG instrumentation. MSG currently keeps track of the allocation of its CPU time among the various MSG jobs (i.e., the control job, the job running the WM, etc.) and among the various functions (e.g., time spent logging, time spent handling status reporting). In its major role of delivering messages and alarms, and processing connection requests, MSG now keeps track of the number of such operations, their various failure counts, and computes their average processing time. Information of this type will be important in determining the effectiveness of the current NSW use of MSG, as well as evaluating alternative MSG utilization strategies. Additionally, MSG now monitors its internal resource allocation strategies, maintaining statistics on the usage counts for its various resources (e.g., tables, semaphores) as well as the frequency of contention for these resources. These measurements will prove valuable in evaluating the necessity for internal reorganization of the MSG program code, and determining the effectiveness of any such reorganization. The MSG statistics are reported via the STATISTICS user command. A sample output is reproduced in Appendix A.

3. The Foreman: Running Tools in the NSW

This quarter we have prepared and distributed an updated version of the Foreman design specification document. The updated report (BBN Report 3442) incorporates all of the design changes and clarifying descriptions which have resulted from building the prototype multi-host NSW. Additionally, we include for the first time the protocol documentation for the inter-component NSW message format, and completely outline the protocol scenarios for beginning and terminating tool sessions. Although parts of these discussions go significantly beyond the domain of the Foreman component, they are crucial to a successful implementation and mark the first time such information has been conveniently available.

3.1 File System Reliability Measures

As part of our NSW system design work, we have, along with project participants from Massachusetts Computer Associates and other NSW contractors, begun to outline a plan for making NSW operation more reliable. The so-called interim reliability plan outlines the major component responsibilities in identifying failures, preserving the integrity of the NSW file catalog, and avoiding the loss of files trapped in a tool's working directory (workspace) during a tool session which is prematurely terminated as a result of a component failure.

Timeouts provide the major device by which components detect possible failures. The interim reliability plan specifies that all component interactions will have an appropriate timeout interval, and further specifies a set of actions to be taken to preserve any tool workspace files which have not yet been fully accounted for by the Works Manager. In brief, the Foreman relevant parts of the reliability plan goes as follows.

Each Foreman instance will monitor its tool's NSW environment using such devices as message timeouts and broken network connection signals. Whenever an FM detects a malfunction that it cannot rectify, it will stop the tool, save the state of the tool's file workspace, and report to a WM that the tool session has been saved. If possible, the FM will alert the user as to the type of malfunction detected. Should there be no WM currently available, the session data is still saved and marked to be reported at a later time. The FM will also be receptive to requests from other components to save the tool session because of some condition that they have encountered which precludes continuing the tool. An example of such a case is the FE losing contact with the NSW user. Additionally, each FM will be required to maintain the tool's local name directory (LND), which describes the state of the local tool filespace, on non-volatile storage. On TENEX this means that the LND is maintained in a secondary storage file which, except in extraordinary circumstances, is normally retained across host system restarts. A crash-proof LND allows for the subsequent reporting and saving of tool sessions which were in progress at the time of any local TBH system malfunction. These postmortem salvaging operations are designed to be performed by an FM process (or its designate) when the TBH is next restarted to run as part of the NSW.

A further measure of safety for the user's NSW files will be provided by periodically checkpointing the NSW file catalog and by the use of WM guarantee messages. To remove the possibility of "losing" newly delivered files due to a WM host crash before the NSW file catalog has been properly saved on a non-volatile storage medium, the FM will delay deallocating any workspace files until it receives a guarantee message from a WM process. The guarantee message is sent to the FM only after the NSW file system descriptor data base has been checkpointed to include all files delivered during the particular tool session. When an FM receives such a guarantee message, the files and their descriptors for the tool session are immune to component and host malfunction (except in rare circumstances). Hence, the FM can safely delete all data associated with the tool session. Naturally, if the FM does not receive the guarantee message within an appropriate timeout interval, it assumes a WM malfunction, and automatically saves the tool session for reporting at a later time.

After a session is reported as "saved" to the WM, the design states that it will be possible for the user to "rerun" the saved tool session. This means continuing the session at least to the extent of allowing the delivery of any saved NSW files and possibly continuing with tool activity. Restarting an interrupted tool session requires the cooperation of a WM, an FM on the affected TBH, and an FE process serving the affected user. Message interactions to implement such a scenario are currently being specified. Once the design has been finalized, we expect to immediately proceed with an implementation. When these reliability mechanisms are in place in the various components, NSW users should be well protected against the possibility of losing completed work which would otherwise be saved had the tool been used outside of NSW. Additionally, the design provides for the possibility of restarting the tool execution within the context of the saved file workspace, a feature not often found in conventional operating systems.

3.2 NSW Message Transmission Convention Changes

At the special contractors' meeting held after the project review at ISI, a number of protocol issues were discussed, and some changes to the encoding of NSW inter-component messages were agreed upon. One important change to the NSW transmission protocol (NSWTP) was to expand the field allocated for reporting error conditions to include an error class code indicating the severity of the error (e.g., temporary failure, error causing process termination, etc.), a textual description of the error suitable for human recognition, and a specific error indicator for program recognition. In the former NSWTP specification, only the specific error code was present in the message header. These changes should make it easier to debug the interactions between components, as well as provide more detailed error information to aid recovery procedures by both NSW users and system component programs.

Another measure which was adopted was the uniform use of NSWB8 data types for all arguments and results within NSWTP. (For a complete description of NSWB8 and NSWTP conventions, see Appendix 3 of BBN Report 3442.) The aim here was to remove (and no longer use) instances in which arguments were described as being one of a number of possible types. In many cases, logically equivalent substitutes which exhibit more structure can be agreed upon. An example is the use of a character string of length zero to represent "no string" instead of an "empty" data type. The uniformity of the data type in a specific argument, regardless of the actual result, makes it easier to write clear, concise code to process and generate NSW messages. It is our belief that these gains outweigh any slight increase in the number of bits needed to represent certain messages.

All contractors agreed that the NSW would switch to these new protocols on December 20, 1976. This necessitated the immediate insertion of these changes into the TENEX Foreman component. The implementation effort proceeded smoothly to meet the deadline.

3.3 Foreman Implementation Improvements

There were a number of improvements made to the TENEX Foreman component during this reporting period. The most extensive implementation change was the conversion to using MSG for establishing direct tool connections to the FE. Prior to this change, the FM and FE directly used their host NCP for initiating connection protocols. As a consequence of this, the FE/FM protocol was intertwined with ARPANET socket protocol. It was intended all along to have MSG mediate the major aspects of the communication needs of the NSW components, including managing the establishment of direct connections. However, in the phased MSG implementation plan, it was not until recently that direct connection handling by TENEX MSG became operational. This was followed immediately by the TENEX FM and TENEX FE protocol specification and implementation to use MSG for their direct connection needs. Following the initial implementation, there were a number of joint debugging sessions involving both BBN and Compass personnel to install and debug the various new components. These inter-contractor sessions have again proven to be of much greater utility than individual debugging sessions using borrowed components. The reason for not having these sessions more often lies in the difficulty in coordinating both the schedules of the people involved and the schedules of the inter-related implementation efforts.

Another prominent Foreman implementation change is the experimental introduction of the use of the MSG "null signal" as a standard interface when sending selected messages. As background for a discussion of the impact of such a change, we first briefly describe the TENEX MSG signalling facility.

Whenever MSG accepts an operation which cannot be immediately completed, it establishes a pending event which is used to notify the process of the actual completion of the operation. The outcome of the operation is reported to the calling process via a memory cell within the parameter block associated with the operation. MSG offers a number of alternatives in helping the application process utilize this stat s in a timely fashion. One mode of operation is to have MSG block the operation of the calling process until the pending event completes. These "unblock" signals lead to a sequential program, meaning that each atomic operation must complete before another can be begun. Using this approach, it is straightforward to program an application process to use MSG primitives. The major drawback is that in some cases, real time performance may suffer with a strictly sequential approach. In those cases where parallel operation is warranted, overlapping MSG processing with further application processing, or remote MSG processing with other local MSG processing can often lead to response time improvements. Accordingly, MSG supports a pseudo-interrupt (PSI) signal which the process can use instead of the unblock signal. Under this mode of operation, control is returned to the calling MSG process after only minimal local validity checking and before the completion of the posted pending event. The process can then logically immediately continue with its processing, and even make other simultaneous MSG requests. To avoid the need for continuous polling of the event status by the MSG user process, a PSI signal is delivered to the process to indicate the completion of a pending event. Thus, by properly configuring the pseudo interrupt system and by providing the programs to handle the interrupt and synchronize the concurrent activities, one can achieve the desired degree of parallelism.

Adding code to handle the parallel activity is not particularly difficult. However, programming an interrupt driven system often results in many difficult timing dependent program bugs. Additionally, there is frequently a non-trivial amount of processing and context switching associated with handling and coordinating the interrupts with the mainstream processing. In view of this, and more importantly, because the higher level NSW intercomponent protocols provide a further measure of error and malfunction detection, we decided to try a different strategy. On an experimental basis, we have adopted a methodology whereby the component initiates parallel activity, but in some cases relies on a subsequent logical event, instead of the MSG PSI signal, to indicate the success or failure of the operation. In particular, in those cases where the FM sends a message to another component and expects a reply, we can use the receipt of the reply as the conclusive proof of the success of the send operation. Also, since the corrective actions taken by an FM on non-receipt of the reply are the same regardless of whether the send operation itself failed or a problem developed after the initiating message had been delivered, a specific failure-to-send signal is not critical. A similar analysis holds for the FM/FE interactions to open their direct connections. The opening of the connection is the status indicator for the correct receipt of the message calling for the connection.

To accommodate such behavior, a "null" signal was added to the set of signals which a process could request for reporting the results of an MSG operation. With the null signal, control is returned immediately to the invoking process, but no further signal (other than changing the parameter block status word) is sent to the process to indicate the operation's completion. Thus, when using the null signal, there need be no further application process overhead after initiating the request. However, we reiterate that this mode of operation is meaningful only in cases where the occurrence of an error is not important because it can be accounted for in some other manner, and where there is some other meaningful activity which can proceed in parallel with the MSG operatio. If no further processing is required, then the unblock signal is just as efficient, yet provides more information.

Since NSW error detection is based on timing out significant expected events, there was no danger of violating protocol in ignoring the success/failure indicator for events which did not require timers. The only drawback to the "null signal" approach would be that certain errors (e.g., those resulting from the unavailability of a remote host) would take longer to be recognized. That is, instead of receiving notification of the failure based on the send operation, notification would be delayed until the receive timeout period had elapsed. This interval reflects both a roundtrip message delay as well as message processing time and is obviously going to be longer than any appropriate MSG based timeout exclusively for the send operation. However, since the error cases are expected to be the exception and not the rule, and since the most common error conditions are those which detach the user from the FM thereby removing any responsiveness issues, we proceeded with the experiment of inserting the null signal as the default in selected MSG operations. We now hope to be able to document the performance improvements resulting from the parallelism and use of the null signal instead of using blocking signals and sequential operation.

3.4 Documentation for Creating Workspace Definition Files

As reported last quarter, we have developed a software module which will, under interactive guidance from an NSW operator, create a new TENEX Foreman workspace definition file. This file, which is used as a common data base by all co-located TENEX FM in a particular NSW configuration, governs the allocation and deallocation of the individual host workspaces dedicated to support NSW tool activity. This quarter we have distributed this software to all NSW participants along with some preliminary documentation to aid its use. (For the record, the documentation is reproduced as Appendix B to this report.) The general release of the program and its documentation now make it possible for each contractor to configure his own TENEX TBH, using resources dedicated to his configuration.

3.5 Additional NSW Tools

This quarter we have made an important addition to the list of NSW supported TENEX tools. FTP, the user program implementing the ARPANET file transfer protocol, has been successfully encapsulated and installed in our test configuration. FTP is a tool which is used to move files between ARPANET hosts. In its NSW encapsulated form, FTP is used to transport files between NSW filespace and the filespace on any ARPANET host supporting the FTP protocol. Using FTP as an NSW tool is similar in function to the NSW EXEC functions for importing and exporting files. That is, the FTP "get" operation is similar to the NSW "import" function, while the FTP "send" operation is equivalent to the "export" function. The major difference is not one of function but rather one of the configuration needed to carry out the operation. NSW file transports require NSW file package jobs on both the transmitting and receiving hosts. To date, there have been only a few implementations for file package processes, and even these are not run as general subsystems. FTP servers, on the other hand, are available for almost all ARPANET hosts and are usually part of the normal operation on these hosts. Hence, using FTP as an NSW tool provides for NSW importing and exporting capability throughout the ARPANET without the need for special NSW software on all of these hosts. Such a capability will become extremely important in any operational NSW, especially when one considers the standard use of special ARPANET host directories in using remote devices such as line printers and tape drives.

4. Meetings and Seminars

In addition to the previously mentioned project review and contractor meeting in December at ISI, there were a number of other meetings and seminars regarding the NSW project. During this quarter we have met with representatives of the Digital Equipment Corporation a number of times regarding the proposed TOPS-20 monitor changes to support NSW operation. We have prepared a document which is the current specification of what is generally agreed to be the prudent changes to the TOPS-20 JSYS repertoire for supporting TOPS-20 as a TBH. These modifications will now once again be examined by DEC personnel, and finalized shortly. After acceptance of the technical specification, there still needs to be an implementation within TOPS-20, along with the subsequent acceptance of the system changes by DEC and a plan for getting these changes into the standard distribution cycle. However, the apparent agreement on the form of the necessary changes marks tangible progress towards a TOPS-20 NSW TBH.

There were also a number of meetings between BBN and the Gagliardi Systems Group (GSG) regarding the use of MSG in illustrating general network interprocess communication capabilities. GSG has been assigned the task of developing demonstrations which illustrate the evolving NSW technology. In addition to demonstrating the NSW itself, their plan was to package a demonstration of the MULTICS interactive data base management system, JANUS, coupled to remote TENEX user programs via MSG. The intent here was to illustrate the feasibility of easily integrating software systems running on host computers which are physically distributed. Furthermore, the coupled systems were built for different types of mainframe computers, and developed from different programming languages. We have been providing consulting services to GSG in specifying the technological areas to be demonstrated and in developing the scenarios which illustrate them. We also have been guiding the implementation of the scenarios as they relate to our TENEX and MSG software.

Finally, this quarter we have given a technical seminar at the State University of New York, Stony Brook on the design of the NSW system and its role in the overall program to improve the methodology of building software systems. This invited guest lecture is indicative of the interest shown by segments of the computer science community in the NSW project. Such lectures also provide a forum for the early exposure of the NSW system concepts to people who are in a position to offer constructive criticism at a time when it can be most meaningfully applied.

Appendix A: Sample MSG Self-evaluation Statistics

The following is a sample of the performance statistics now accumulated by MSG, and reportable via the MSG STATISTICS command.

```
  Statistics for MSG Version  1 Incarnation  927 at BBN-TENEXB
   Started 12-Dec-77 13:59:26  Last update at 15:31:06.104

   MSGs Started =6 MSG CPU =1:4.576 in 7:5:46.958
   Procs Started =13 Top CPU =58.624 All CPU =2:18.321 in 2:18:53.309
 Control CPU =0.000 Total Job CPU =6:8.276 (Difference =2:45.379)
 CPU used in RUP =2.211 Logging =0.000 Status Reporting =1.883

   User primitives issued =299   Signals rcvd by MSG processes =137
 User Prim Failures =0 PE Failures =1 PE Timeouts =1 ICP Failures =826918034

   Messages delivered =111 Total Delay =11:49.956 Avg Delay =6.396
   Alarms delivered =0 Total Delay =0.000 Avg Delay =0.000
   Connection Opens/Closes =17 Total Delay =3:2.506 Avg Delay =10.735

   Semaphore Locks
     Usages Contentions  Function
       1294           1  Bit Tables
        602           0  Job Control Blocks
        810           0  Process Control Blocks
        140           1  Host Control Blocks
         89           0  Generic Table Entries
        232           2  Transaction Control Blocks
        916          12  Other:
        576           7    Timer Queue
         34           0    New Semaphore
        305           5    RUP List
          1           0    Logging
 Bit     Num    Max    Max    Lock    Lock
Table   Alloc  Alloc  Avail   Uses Contentions
Conns       3      3     32      3      0
Forks      13     13     36     13      0
Socks       3      3    192      4      0
JCBs        7      7     16      8      0
PCBs       10     10    128     20      0
PEs        22     28   1024    504      0
TCBs       12     18   1024    256      0
SQs         0      1    512    134      0
MBs         7     13    512    213      0
SSMs       24     24     24     57      0
DSMs       33     33    192     40      0
RNMs        0      0    256      0      0
Hs          5      5     32     12      1
GNs         5      5     20      5      0


   User  Primitives Issued (PE producing)
  SSM   SGM   RSM   RGM   SAL   EAL   OCN   CCN
    9    14    30     0     0     0     5     3  FE
   31    16    22    27     0     0     0     0  WM
   11     0     0    12     0     0     0     0  FLPKG
   13    12    30     7     0     7     5     4  FOREMAN
    3     0     0     1     0     0     0     0  CHKPTR

 Current time is 15:32:40.227 or (really) 12-Dec-77 15:32:40
```

Appendix B: Preliminary Documentation for the MKCOM Program

The following is the preliminary documentation for using the utility program for making a Foreman workspace definition file.

The program name is MKCOM.SAV. It should be run to create a new TENEX Foreman workspace definition file, or to modify an old one. (Currently, we support only creating a new file.) The program initially asks if a new file is to be created. Answer "Y" or "y" to create a new file (i.e. no previous information), or "N" ("n") to update the information in an existing file.

For creating a new file:

The program will ask for a workspace directory name. Type (upper or lower case, doesn't matter) the name of a directory to be used by NSW as a TENEX workspace on that host. Do not use punctuation to delimit the name (i.e. no < or >). Example:
    NSW-WSD1 A carriage return (eol) delimits the end of the workspace name. A line which contains only a CR means there are no further workspaces to be added. The program will immediately verify that the workspace name given is indeed valid, and if it is will proceed to obtain its password. If the workspace name is invalid, it will repeat the question asking for the workspace name. You can now try another name. There are currently no line editing characters. If a typing mistake is made, make sure the typed input is not a valid string (perhaps by adding an out-of-band character) and type CR, which will cause the tests to fail, and have the question repeated.

When the program asks for a password, type either the password string, or null line (just CR). If the password string has been typed (upper case and lower case ARE distinguishable here; in all probability the directories you are concerned with will be all uppercase) then the program verifies that it is correct. If not correct, the question will be repeated, and you can try again. Correct mistakes as above. By typing a null line in response to the Password request, you are indicating that no password is necessary for the Foreman to connect to that workspace. We then will rely on the group relationship to be properly setup between the login directory of the job running the Foreman and the workspaces.

After you have indicated that there are no more workspaces to be added (by typing a blank line in response to the request for another workspace) there will be a series of question relating to which of the directories in the set you wish to be used actively. The theory here is to have a "common" file listing all the workspaces, and to have individual copies of the common file indicate in the header of the file a subset of the workspaces to be used by the Foreman executing off of that file. Workspaces are assumed to be indistinguishable, so the questions relate to the number (index) of the workspace beginning the usable section, and how many workspaces (starting from the given beginning) can be used by the executing Foreman. The answers to these questions should be decimal integers, terminated by a CR. Validity checks are made. Optionally, a blank line (CR only) can be typed when asked for the beginning of the section. This indicates that you wish to use all available workspaces when you give the file to the executing Foreman.

example:
Say we have defined 10 workspaces (logically number 0 thru number 9; note that these indices have nothing whatever to do with the actual workspace name e.g. NSW-WSD9, and that the actual workspace names need not have any numbering in them at all. The first one you type is 0, the second 1, etc., again assuming all workspaces to be equivalent. If we wanted to produce a workspace definition file which told the executing Foreman to use a block of workspaces which consisted of the last 4 workspaces in our list, we would say beginning=6, length=4. Presumably, another file will be created (using the update facility) which will be used to drive another concurrent Foreman coexisting on the same host, which will use some/all of the remaining workspaces.

When the program is satisfied with your answers to these questions, (again an inappropriate answer results in the question being repeated) it will ask for the name of the new file. You will now be talking to TENEX GTJFN, and should give an appropriate file name (TENEX editing characteristics are in effect here). Completion and confirmation are used. The program indicates when it is complete.

The name of the actual file to be used by the executing Foreman MUST be FORCOMFILE.SHR (highest version will be used), and this file MUST exist in the Login directory of the Foreman NSW job. (e.g. NSWTST on BBNB, NSWDM on ISIC).

In running the program, you can call the output file anything you want, and place it anywhere you want. To run a Foreman, you must have copied, renamed, or initially created a file with the appropriate name and put it in the appropriate place. A new file should be created for each TENEX host on which we have a Foreman, and workspace definition files should NOT (yet) be moved from one host to another.

We are currently using 10 workspace directories, named NSW-WSD1 thru NSW-WSD10, with the appropriate passwords. These exist/should exist on the BBNB, BBNE, ISIC, and ISID systems and are to be used exclusively by the TENEX NSW Foreman. Take care in placing names as potential workspace directories, as these directories will very frequently have ALL of their files deleted and expunged.